

SYNCHRONOUS AND ASYNCHRONOUS COLLABORATION BETWEEN HETEROGENEOUS APPLICATIONS

TECHNICAL FIELD

[0001] The present invention relates to collaboration between applications and more particularly, to software that provides synchronous and asynchronous collaboration between heterogeneous design applications.

BACKGROUND INFORMATION

[0002] Collaboration on a common project by people at different locations has become more common. Indeed, companies may have engineers from many different geographical locations working as a team on a common project. To collaborate on the project, the individual members of the engineering team either travel to a common location or use computer networks.

Historically, when using computer networks to collaborate, a single engineer will take the lead and transmit drawing images showing potential changes to the product design to other members of the team. The computer-network system of collaboration can be more efficient than attempting to bring all of the members of the collaboration group into a single conference room to crowd around a table and review a single design drawing as the group leader makes changes.

[0003] In certain collaborative projects, such as the design and manufacture of electro-mechanical products, all team members are generally working towards design and manufacture of the same product. However, the needs of their specific domain expertise may require each to use a different design abstraction (e.g., schematic for functionality, floor planner for constraint tradeoffs, layout for placement and routing, mechanical for design enclosures, DFM for conformance with manufacturing processes, etc.). This creates unique challenges when collaborating over a network because each participating domain expert may potentially be using a different application during a collaboration session and yet all want to simultaneously reference the same design.

[0004] Referring to FIG. 1, one example of a current approach to network-based design collaboration is shown. According to this approach, a single design application 10 is located on

either a server 12 or a client 14 connected to a network 16. In the server-based implementation, the application 10 is hosted on a server 12 and all other participating clients 14 use the network (e.g., the Internet) to connect to it via a session hosted on the same server. One example of this approach has been implemented by Citrix Systems, Inc. In the client-based implementation, the application 10 is hosted on one of the clients 14 and all of the other participating clients 14 use the network 16 to connect to it via a session hosted on a remote server 12. One example of this approach has been implemented by WebEx Systems, Inc.

[0005] The server 12 includes a session manager 20 for managing connectivity and data traffic including transfer of the application images between the clients 14. Each of the clients 14 includes a session client process 22 that synchronizes user interaction and image display between the local client and the session. During a collaboration session, there is a flow of messages 24 for session controls that include the application images. Regardless of where the controlling design application 10 is hosted, each client 14 is allowed to interact with the application 10 in a way that enables other participants to see what is being done.

[0006] The above approach to network-based design collaboration has limitations. Because each user shares a single image from a single application, this approach is limited when the design activities involve simultaneous usage of different or heterogeneous design applications (e.g., different CAD tools). The sharing of a single application also presents licensing issues. The user license for the application may not allow multiple users of the application over the network, resulting in potential licensing violations. Also, the transmission of the actual images over the network presents security concerns. The unauthorized interception of messages exchanged during a collaborative session may result in the disclosure of confidential design drawings.

[0007] Accordingly, there is a need for a collaboration system and method that allows heterogeneous applications (e.g., different CAD tools) to be used and that does not require transmission of the design images.

SUMMARY

[0008] In accordance with one aspect of the present invention, a computerized system and method is provided for synchronously collaborating over a network to manipulate a design using a plurality of heterogeneous user applications running concurrently on respective clients connected to the network. The system and method involve connecting a session client process to

a session manager over the network to participate in a collaborative session and sharing session control messages with other session client processes connected to the session manager. Design data representing the design is loaded into a local application running on the client and at least one application state file is created representing at least one application state of the local application based on manipulation of the design using the local application. The application state file is dynamically communicated from the session client process to the other session client processes via the session manager. Application state files created by other local applications and communicated from the other session clients via the session manager are loaded into the local application.

[0009] According to another aspect of the present invention, a computerized system and method is provided that can be used for asynchronous collaboration. This system and method involves loading design data into a local application on a computer and manipulating the design using the local application. One or more application state files are created representing an application state of the local application based on at least one manipulation of the design using the local application. The application state file(s) are saved in a journal file and the journal file is transmitted to another computer at some later time such that the application state file(s) can be loaded on the other computer at some later time and the manipulations can be reviewed using another heterogeneous application running on the other computer.

[0010] According to another aspect of the present invention, a computerized system and method is provided that can be used for dynamically collaborating between at least two heterogeneous applications running concurrently on a single client. This system and method involves loading design data into each of the applications. Every time a design object is manipulated (e.g., an object is highlighted) in one of the applications, an application state file is created that reflects the name of the object highlighted and the other application is dynamically notified of the application state file using a local operating system inter-process messaging. The other application can then read this application state file and similarly manipulates a corresponding design object in its database based on the application state file. This process is preferably bi-directional.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] These and other features and advantages of the present invention will be better understood by reading the following detailed description, taken together with the drawings wherein:

[0012] FIG. 1 is a functional block diagram of an existing network-based collaboration system.

[0013] FIG. 2 is a functional block diagram of a network-based collaboration system, according to one embodiment of the present invention.

[0014] FIG. 3 is a functional block diagram of the client and server in a network-based collaboration system, according to one embodiment of the present invention.

[0015] FIG. 4 is a flow chart illustrating a synchronous collaboration method, according to one embodiment of the present invention.

[0016] FIG. 5 is a flow chart illustrating an asynchronous collaboration method, according to another embodiment of the present invention.

[0017] FIG. 6 illustrates one example of an application state file.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0018] Referring to FIG. 2, a network-based synchronous collaboration system and method is used to collaborate on a design using heterogeneous applications 30-36. The applications 30-36 are heterogeneous in that they use different design abstractions and different local data models. Collaboration in the context of this invention means synchronous or asynchronous sharing of design data among one or more computers for the purpose of creating, modifying, annotating, reviewing, documenting or otherwise manipulating the design data. The system and method enables design collaboration between the heterogeneous applications 30-36 by exchanging application states normalized at a design abstraction level recognized by each of the heterogeneous applications 30-36. Thus, the heterogeneous applications 30-36 can be used to collaborate on a design without having to transmit the design images between the applications. Although the network-based synchronous collaboration system and method is especially suited for use in the context of computer-aided design (CAD) applications employed in the design of electronic and mechanical assemblies, the system and method can be used in any collaborative environment and with any type of applications.

[0019] According to the network-based synchronous collaboration system and method, a server 12 and any number of clients 14 are connected to a network 16. A session manager 40 is located

on the server 12 and session client processes 42 are located on each of the clients 14 and integrated with the respective local user applications 30-36. The session manager 40 manages connectivity and data traffic 46 between the session client processes 42. The data traffic 46 includes session messages for session control, i.e., normalized instructions for the session client processes 42. The data traffic 46 also includes the application states that convey the state of the individual applications 30-36, which are then interpreted by each application 30-36 in its own context. The data traffic 46 preferably does not contain any images and the transmission of images is not required for the collaboration, although the present invention does not necessarily preclude the transmission of images together with the application states.

[0020] Although the exemplary embodiment is an internet-based synchronous collaboration system and method, the present invention can also be implemented on other types of networks including, but not limited to, intranets or extranets. In another embodiment, the invention can be implemented on a stand-alone computer, for example, in an asynchronous mode.

[0021] The application states are preferably encoded with a set of normalized XML structures designed for use by different applications via built-in interpreters. Instead of encoding entire images, these XML structures refer to states and database objects on an abstraction level common to the involved applications 30-36. In other words, the applications 30-36 communicate their states with respect to their databases, instead of communicating the images. In CAD applications, for example, the application states can include zoom, pan, units, visibility, color, highlights, and the like, the design objects can include layers, classes, components, pins, nets, test points, object attributes and properties, and the like and the annotations can include text, drafting, URL hyper links, and the like. One example of an application state file in XML format is shown in FIG. 6.

[0022] The ability to convey application states via a normalized set of XML messages allows sharing of interactive events, such as the highlighting of an object across all applications capable of referencing such object. Because the heterogeneous applications 30-36 can interpret the application states, the users are not limited to using a single application sharing the same image during a collaborative session. Security is enhanced when the actual images are not exchanged during the live on-line collaboration, thereby avoiding the need for encryption to protect unauthorized interception of images.

[0023] User applications running locally on a client also may avoid the licensing issues involved with multiple users simultaneously interacting with a single application over a network.

Licensing violations are less likely to occur when each user is using a local application for which the user is licensed. Also, an architecture of distributed heterogeneous applications running on separate clients improves interactive performance of the collaborative sessions by decreasing the processing load on the server. Interactive performance is also improved by eliminating the need to transfer graphic images.

[0024] These XML structures are preferably defined and communicated through external files, which are not dependent on the control and flow of the collaborative session. The function of the session manager 40 is preferably limited to managing client connectivity and exchanging the application states between clients 14. This separation of collaborative session controls from details of application states provides flexibility. For example, the user can select or upgrade collaborative services (i.e., providers of the server-based session manager 40) and the change of service provider is less likely to limit collaborative choices in specific design domains.

[0025] Referring to FIG. 3, one embodiment of the network-based synchronous collaboration system is described in greater detail. Each client 14 includes a user application 30 and a session client process 42 for synchronizing interaction between the user application 30 and the collaborative session. Each client 14 replicates generally the same integration with its session client 42, but the application 30 can be different from applications on other clients 14. An application state file 52 created on the client 14 captures the application states of the user application 30. The application state file 52 is preferably in XML format and a domain specific XML dictionary (i.e., a DTD) defines XML constructs, keywords and tags used by the application state file 52 for a particular design abstraction. For example, for printed circuit board designs, an application state file for a schematic abstraction is based on a different XML dictionary than an application state file for a physical layout abstraction. Similarly IC designs use different dictionaries from printed circuit board designs. The user application 30 preferably includes a reader/writer 80 for reading from and writing to the application state file 52. One example of an application state file 52 is shown in FIG. 6. A session journal file 54 stored on the client 14 maintains a log of the session activities.

[0026] The session client 42 includes a user interface 60 for user interaction with the user. The client 14 can display the user interface 60 on the same screen 61 as the user interface 63 for the

application 30. The user interface 60 preferably provides a consistent look and feel across multiple clients 14, operating system platforms and user applications 30.

[0027] The user interface 60 allows the user to control and monitor various functions and interactions of the session client process 42 and the local user application 30. The user interface 60 can be used to initiate and terminate the local user application 30 and to locate and load appropriate design files into the user application 30. The user interface 60 can also be used to schedule new sessions and to log in to and out of scheduled sessions via the session manager 40. The user interface 60 displays all communication activity with the user application 30 and with the session manager 40. The user interface 60 can be used to provide chat style textual communications among the active session participants in a chat line like form. The user interface 60 allows the user to control sending and receiving the application state file(s) 52 to and from other session participants. The user interface 60 also allows the user to control initiating the reading and writing of the application state file 52 by the user application 50. The user interface 60 further allows the user to control writing to the local session journal file 54 and initiating play-back of the session journal file 54.

[0028] The client session 42 includes external controls and application programming interfaces (APIs) 64, 68 for synchronizing interactions with the session manager 40 and the local user application 30. Session synchronization between the session client 42 and the session manager 40 is provided through session controls 62 passed via the appropriate industry standard Internet communication channels. The session controls 62 are defined and enforced by the session client 42 and they instruct the session clients 42 and the session manager 40 as to what to do next with respect to collaborative activities. The session controls 62 also transport the application state file(s) 52 to other session clients 42 that participate in the collaborative session. Session synchronization between the session client 42 and the user application 30 is provided through a set of application controls 66 passed via the related application programming interface (API) 68. The application controls 66 are defined and enforced by the session client 42 and they instruct the session client 42 and the application 30 as to what to do next with respect to the collaborative activities. Although the technical details of such controls depend on the operating system, two examples are OLE Automation in the Microsoft Windows operating system and sockets in the Unix operating system.

[0029] The session client 42 also includes internal controls for handling interaction with the application state file 52 and the journal file 54. Application state file controls 70 transport one or more application state file(s) 52 to and from the session manager 40. Session journal controls 72 record session activities in the session journal file 54 and communicate the content of the session journal file 54 to the user application 30.

[0030] The user application 30 can be based on an existing design application, which has been designed or extended to properly read, write and interpret the XML structures and to integrate with the session client 42 through the controls 66 and API 68. One example of an application that can be used is the application (and its API) available under the name InterComm from OHIO Design Automation Inc., which is described in co-pending Patent Application Serial No. 09/885,834 incorporated herein by reference. Other examples include the applications available under the names Allegro and Concept from Cadence Design Systems Inc. and the SKILL API used with these applications.

[0031] The session manager 40 can be based on an existing session manager, which has been designed to handle session controls 62 and to integrate with the session clients 42. One example is the session manager used in the Internet communications services provided by WebEx Communications, Inc.

[0032] Referring to FIG. 4, one method of synchronous collaboration is described. A user can schedule a new collaborative session using the services of the session manager 40, step 110. The user can log in to the session manager 40 with a username/password, schedule a new collaborative session, and invite participants. The user can also communicate to the session participants the process for obtaining the appropriate design databases. For example, each session participant may load its own version of the design database compatible with the local application using local PLM procedures. The scheduling step can be performed using the session client 42 or using an Internet browser, depending upon the implementation. Access to the services of the session manager 40 can also require a subscription based on fees.

[0033] On the scheduled day/time, the user and other invited users log into the session manager 40 using the user interface 60 of the client session 42, step 112. Session login may require the user to enter a previously defined username and password.

[0034] Once the session manager 40 determines that the collaborative session was initiated and that the appropriate parties have joined, it starts sharing session controls (including information

as to the origin of the controls) received from one session client with all participating session clients, step 114. All local session client controls exchanged with the user application 30 and/or the session manager 40 are preferably recorded in the local session journal 54 of each client 14.

[0035] The user initiates the local application 30, step 116, and then locates and loads the related design data into the user application, step 118. Depending on the details of the local integration between the session client 42 and the user application 30, these steps can be initiated either through the session client or through the user application 30. Also, these steps can be performed by the local application 30 either before or after logging into a collaborative session. The local application 30 can then be used to manipulate the design, step 120.

[0036] According to the user session priority and privileges enforced by the session manager 40, one or more of the users can use the session client 42 user interface 60 to start a chat line like text-based conversation with other participants, step 122. This text is sent to the session manager 40, which in turn distributes it to all participating session clients 42 for display through the respective user interfaces 60 along with the information that identifies the originator. All chat line messages sent and received are preferably recorded in the local session journal 54 of each client 14.

[0037] According to the user session priority and privileges enforced by the session manager 40, one or more of the users can use the session client 42 user interface 60 to communicate application states to the others, step 124. In particular, the user application 30 is instructed by the session client 42 to create an application state file 52 (e.g., using XML structures) based on design manipulations, and the user application 30 informs the session client 42 that the application state file 52 was generated. The session client 42 sends the application state file 52 with the appropriate session controls to the session manager 40. The session manager 40 distributes the received application state file 52 with the appropriate session controls to all active participants. All application states are also preferably recorded in the local session journal 54 of each client 14.

[0038] When an application state file 52 is received, session clients 42 send a session control 66 to the local user application 30 indicating that a new application state file is available. The local user application 30 loads the newly arrived application state file 52 using parser 80 to reflect a new state of the display. Using local session client UI 60, the user can control when the synchronization actually occurs and how long it is in effect, without affecting active connectivity

with the collaborative session. While the session client 42 continuously buffers application states files 52 from the other participants, for example, users may refuse or delay instantiation of the newly arrived application state file 52 in their user application 30. This on-demand synchronization allows the user to work on something independently with the user application 30 while the collaboration continues and then catch up later without losing the details of the concurrent collaborative exchange.

[0039] The collaboration continues until all participating users log out of the session using their local session clients 42, step 126. The session clients 42 can reload the local session journal files 54 anytime after the session has been terminated in order to replay the entire collaborative session in an asynchronous mode.

[0040] Users also have an option to use the session client 42 to create a session journal 54 file without ever logging into a collaborative session. This is a form of an asynchronous collaboration. Journal files 54 can also be generated entirely outside of the session client 42 through any appropriate means and then shared with session clients 42 in an asynchronous mode.

[0041] Referring to FIG. 5, one method of asynchronously collaborating is described in greater detail. The method is similar to the synchronous collaboration method described above, without the steps of scheduling and logging in to a collaborative session and sharing session controls.

The user initiates the local user application 30, step 210, and the design data is loaded into the local user application, step 212. The design is manipulated using the application, step 214, and the application state file is created based on the design manipulations, step 216. The application states files 52 (e.g., XML messages) are recorded in the journal file 54, step 218, which can later be communicated with another user, step 220, and loaded to replay the asynchronous collaboration session without being connected to the server. This allows the user to review the design independently of the other collaborating users. The asynchronous mode can also be used when a synchronous collaboration cannot be scheduled (e.g., due to time zone conflicts).

Asynchronous collaboration can also be used to author instructions such as training, rework or Engineering Change Orders (ECO) for electronic or mechanical assemblies.

[0042] According to another method, collaboration is provided between two heterogeneous applications running on the same client (not shown). This involves loading appropriate design data into each of the two user applications such as Allegro available from Cadence Design Systems Inc. for PCB designs and DX Designer available from Mentor Graphics Corp. for

schematic designs while both applications are running concurrently on the same client. In one example of this method, a design object is modified by highlighting the design object using one of the user applications. Every time an object is highlighted in one of the user applications, an application state file is created that reflects the name of the object highlighted and the other application is notified of this fact using a local operating system inter-process messaging, e.g., OLE Automation on Microsoft Windows. The other user application then reads this application state file and highlights the corresponding object in its database. This process is preferably bi-directional between the user applications.

[0043] Embodiments can be implemented as a computer program product for use with a computer system including, but not limited to, a PC or a mobile device. Such implementation may include a series of computer instructions fixed either on a tangible medium, such as a computer readable medium (*e.g.*, a diskette, CD-ROM, ROM, or fixed disk) or transmittable to a computer system, via a modem or other interface device, such as a communications adapter connected to a network over a medium. The medium may be either a tangible medium (*e.g.*, optical or analog communications lines) or a medium implemented with wireless techniques (*e.g.*, microwave, infrared or other transmission techniques). The series of computer instructions embodies all or part of the functionality previously described herein with respect to the system. Those skilled in the art should appreciate that such computer instructions can be written in a number of programming languages for use with many computer architectures or operating systems. Furthermore, such instructions may be stored in any memory device, such as semiconductor, magnetic, optical or other memory devices, and may be transmitted using any communications technology, such as optical, infrared, microwave, or other transmission technologies. It is expected that such a computer program product may be distributed as a removable medium with accompanying printed or electronic documentation (*e.g.*, shrink wrapped software), preloaded with a computer system (*e.g.*, on system ROM or fixed disk), or distributed from a server or electronic bulletin board over the network (*e.g.*, the Internet or World Wide Web). Of course, some embodiments of the invention may be implemented as a combination of both software (*e.g.*, a computer program product) and hardware. Still other embodiments of the invention are implemented as entirely hardware, or entirely software (*e.g.*, a computer program product).

[0044] While the principles of the invention have been described herein, it is to be understood by those skilled in the art that this description is made only by way of example and not as a limitation as to the scope of the invention. Other embodiments are contemplated within the scope of the present invention in addition to the exemplary embodiments shown and described herein.

Modifications and substitutions by one of ordinary skill in the art are considered to be within the scope of the present invention, which is not to be limited except by the following claims.